04205886 **Image available**

DEBUGGING SYSTEM FOR DISTRIBUTED INFORMATION PROCESSING SYSTEM

PUB. NO.: 05-197586 **JP 5197586** A]
PUBLISHED: August 06, 1993 (19930806)
INVENTOR(s): KAWABE SHIGEHISA
YAMASHITA ICHIRO
APPLICANT(s): FUJI XEROX CO LTD [359761] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 03-060781 [JP 9160781]
FILED: February 27, 1991 (19910227)
INTL CLASS: [5] G06F-011/28; G06F-015/16; G06F-015/16
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units); 45.4 (INFORMATION PROCESSING -- Computer Applications)
JOURNAL: Section: P, Section No. 1646, Vol. 17, No. 624, Pg. 13, November 17, 1993 (19931117)

ABSTRACT

PURPOSE: To easily execute debugging without being conscious of a remote procedure call in a distributed information processing system where plural processes make communication by means of the remote procedure call and realize a parallel operation.

CONSTITUTION: Plural satellite debugging means 3 and 4 which are connected to plural processes 1 and 2 and execute debugging, and a center debugging means 5 which remotely controls the operation of the plural satellite debugging means 3 and 4 are provided. The center debugging means 5 has a remote procedure call issue detecting means 51 which previously detects that there is the remote procedure call in the process of a debugging object in the satellite debugging means 3 (or 4), a remote procedure call end detecting means 52 detecting the end of the processing of the remote procedure call and a switch control means 53 which switches and controls the starting state of the satellite debugging means 3 and 4 in response to the detection of the remote procedure call issue detecting means 51 or the detection of the remote procedure call end detecting means 52.

(19)日本国特許庁（ＪＰ）　(12)特　許　公　報（Ｂ２）　(11)特許番号

第2782971号

(45)発行日　平成10年(1998)８月６日　(24)登録日　平成10年(1998)５月22日

(51)Int.Cl.⁶	識別記号	ＦＩ	
Ｇ０６Ｆ　11/28		Ｇ０６Ｆ　11/28	Ｊ
15/16	３７０	15/16	３７０Ｎ
	４５０		４５０Ｚ

請求項の数１(全:　:)

(21)出願番号　特願平3－60781

(22)出願日　平成３年(1991)２月27日

(65)公開番号　特開平5－197586

(43)公開日　平成５年(1993)８月６日

審査請求日　平成８年(1996)10月18日

(73)特許権者　000005496
富士ゼロックス株式会社
東京都港区赤坂二丁目17番22号

(72)発明者　川邉　恵久
神奈川県海老名市本郷2274番地　富士ゼロックス株式会社海老名事業所内

(72)発明者　山下　一郎
神奈川県海老名市本郷2274番地　富士ゼロックス株式会社海老名事業所内

(74)代理人　弁理士　岩上　昇一　(外１名)

審査官　深沢　正志

(56)参考文献　特開　平１－92847（ＪＰ，Ａ）
特開　平１－147639（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.⁶，ＤＢ名)
G06F　11/28

(54)【発明の名称】　分散型情報処理システムのデバッグ方式

(57)【特許請求の範囲】

【請求項１】　ネットワークで互いに接続された複数の計算機上で、複数のプロセスが遠隔手続呼出しにより互いにネットワーク通信を行い並行・並列動作を実現する分散型の情報処理システムのデバッグ方式において、

前記複数のプロセス各々の実行される計算機上で、各プロセスのデバッグを行う複数の衛星デバッグ手段と、

その複数の衛星デバッグ手段の動作を遠隔制御する中央デバッグ手段とを備え、その中央デバッグ手段は、

衛星デバッグ手段のデバッグ対象のプロセスに遠隔手続呼出しがあることをその遠隔手続呼出しの発行の前に検出する遠隔手続呼出し発行検出手段と、

前記遠隔手続呼出しで呼び出されたプロセスの終了を検出する遠隔手続呼出し終了検出手段と、

前記遠隔手続呼出し発行検出手段の遠隔手続呼出しの検出に応答して、その遠隔手続呼出しにより呼び出されるプロセスを有する計算機上の衛星デバッグ手段を起動すると共に、前記遠隔手続呼出し終了検出手段のプロセスの終了の検出に応答して、そのプロセスの呼出し元の衛星デバッグ手段によるデバッグを再開する制御手段とを有することを特徴とする分散型情報処理システムのデバッグ方式。

【０００１】

【発明の詳細な説明】

【産業上の利用分野】本発明は、複数の計算機を互いに関連制御することによって、複数のプログラムを分散して処理する分散型の情報処理システムに関し、特にこうしたプログラムの並行、並列あるいは分散処理に際してそのシステム内で並行・並列動作する各プロセスを対象として好適なデバッグ方式に関する。

【０００２】

【従来の技術】近年、ネットワークの普及や計算機の低価格化と、計算機に要求される処理の複雑化と多様化により、複数のプロセス各々について通信チャネルを介した通信手段を通じて互いに通信を行い、協調して並行・並列動作するプロセスからなるプログラムが記述されるようになっている。特に、広域ネットワークや狭域ネットワークで結合された複数の計算機に分散配置された前記並行・並列動作するプロセスからなる分散プログラムのプログラミングが行われるようになっている。このようなプログラミングにおいては、協調して並行・並列動作するプロセスを容易に記述する方法やデバッグする方法の開拓が大きな課題となっている。

【０００３】協調して並行・並列動作するプロセスを容易に記述する方法の一つとしては、プログラムを、処理を提供するサーバと処理結果を利用するクライアントとに機能分離を行い、記述することによりモジュール化しプログラミングを行う方法がある。特に、サーバとクライアントの機能分離の方式の一つに遠隔手続き呼出しとして知られる方法がある（たとえば、「分散コンピューティング環境の最前線」、日経エレクトロニクス、１９９０，６，１１、No．５０２、ｐｐ１２２－１４８参照）。遠隔手続き呼出しにおいては、処理の単位を手続きとして捉え、同一あるいは遠隔の計算機上のプロセスに対し手続きを指定し引数を送信し、手続きの返り値を受信する。また手続きの副作用が情報処理システムの状態に反映されることで、処理が行われる。この遠隔手続呼出しを導入することにより簡単に協調して並行・並列動作するプロセスからなるプログラムのプログラミングが可能である。デバッグに際しても簡単に協調して並行・並列動作するプロセスをいかに扱うかが大きな問題の一つとなっている。従来のデバッグ方式は単一のプロセスを扱うにすぎなかったが、近年では並行・並列動作するプロセスを扱うことが可能なデバッグ方式も提案されている（たとえば、(1) D. Caswell、D. Black「Implementing a March Debugger For Multithreaded Applications」Proc. of USENIX, Winter '90, Nov. 1989 pp 25-39、(2)「Domain Distributed Debugging Enviroment Reference」Apollo ComputerInc., Order No. 011024-A00、(3) 特開平１－２００５６３号公報等参照）。また、遠隔手続呼出し機能に通信障害の発生や遠隔手続呼出しのに関する再試行の情報を管理する機構を組み込み、遠隔手続呼出しのデバッグを補助するデバッグ方式も提案されている（たとえば、R. Cooper「Pilgrim: A Debugger for Distributed Systems」Proc. of 7th Int Conf. on Distributed Computing Systems, pp. 458-465, Sep. 1987参照）。

【０００４】

【発明が解決しようとする課題】しかし、従来のプログラムデバッグ方式では、並行・並列動作するプロセスを

デバッグ可能とする機能を重点として考えられており、特に遠隔手続呼出しを用いた並行・並列動作するプロセスのプログラムをあたかも単一プロセスのプログラムと同様に簡単にデバッグすることは困難であるという不具合があった。本発明は、前記従来技術のプログラムデバッグ方式の持つ不具合に鑑み、遠隔手続呼出しを用いて互いに通信を行う複数のプロセスにより実現される情報処理システムにおいて、遠隔手続呼出しを意識しないで簡単にデバッグができるプログラムデバッグ方式を提供することを課題とする。

【０００５】

【課題を解決するための手段】本発明は、上記課題を解決するために、図１に示すように、ネットワークで互いに接続された複数の計算機上で、複数のプロセス１，２（図２の２０３又は２１３で実行されるプロセス）が遠隔手続呼出しにより互いに通信を行い並行・並列動作を実現する分散型の情報処理システムのデバッグ方式において、前記複数のプロセス１，２の各々の実行される計算機上で、各プロセスのデバッグを行う複数の衛星デバッグ手段３，４（図２の２０、２１）と、その複数の衛星デバッグ手段３，４の動作を遠隔制御する中央デバッグ手段５（図２の２２）とを備えている。そして、その中央デバッグ手段５は、衛星デバッグ手段３（又は４）のデバッグ対象のプロセスに遠隔手続呼出しがあることをその遠隔手続呼出しの発行の前に検出する遠隔手続呼出し発行検出手段５１（図２の２２４）と、前記遠隔手続呼出しで呼び出されたプロセスの終了を検出する遠隔手続呼出し終了検出手段５２（図２の２２５）と、前記遠隔手続呼出し発行検出手段５１の遠隔手続呼出しの検出に応答して、その遠隔手続呼出しにより呼び出されるプロセスを有する計算機上の衛星デバッグ手段を起動すると共に、前記遠隔手続呼出し終了検出手段のプロセスの終了の検出に応答して、そのプロセスの呼出し元の衛星デバッグ手段によるデバッグを再開する制御手段５３（図２の２２２）とを有する。

【０００６】

【作用】上記構成において、第１の衛星デバッグ手段３によりデバッグが行われているひとつのプロセス（以下、クライアントプロセス）１が遠隔手続呼出しを発行するとき、遠隔手続呼出し発行検出手段５１はその遠隔手続呼出しの発行を検出するとともに、その要求される遠隔手続を含む他のプロセス（以下、サーバプロセス）２を特定する。中央デバッグ手段４は要求される遠隔手続を含むサーバプロセス２が動作している計算機上に第２の衛星デバッグ手段５を起動させる。また、切り替え制御手段５３は、中央デバッグ手段５と衛星デバッグ手段との間の通信を、第１の衛星デバッグ手段３から第２の衛星デバッグ手段４に切り替えることにより、クライアントプロセスのデバッグを行う者はサーバプロセス２のデバッグをクライアントプロセス１のデバッグに引続

き行う。上記遠隔手続の実行が終了すると、遠隔手続呼出し終了検出手段５２は遠隔手続呼出しの実行の終了を検出する。切り替え制御手段５３は、第２の衛星デバッグ手段４に、サーバプロセス２のデバッグを終了するように指示する。また、サーバプロセス２のデバッグが終了すると、第２の衛星デバッグ手段４の動作を終了させる。そして、第１の衛星デバッグ手段３にデバッグを行わせるよう起動状態を切り替える。これにより、サーバプロセス２のデバッグを行う者はサーバプロセス２のデバッグから引続きクライアントプロセスのデバッグを行うことができるものである。以上のように、本発明は、サーバプロセスとクライアントプロセスの衛星デバッグ手段を適切に制御することができるので、デバッグをする者には、遠隔手続呼出しが発行されたときの通信手順や引数の通信部分、あるいは、遠隔手続呼出しが終了したときの通信手順や返り値の通信部分は見えなくて、あたかもクライアントプロセスから単なるシングルプロセス上の、遠隔手続ではない手続呼出しが実行され、返り値が返されたかのように見える。従って、デバッグをする者は、従来の単一プロセスに対するデバッグ手法を用いることができ、プロセスの並行・並列動作を意識することなく簡単に遠隔手続呼出しを行う並行・並列分散プログラムのデバッグをすることができる。

【０００７】

【実施例】以下、図面を参照しながら実施例に基づいて、本発明の特徴を具体的に説明する。図２は、本発明の実施例を示すブロック図である。図２において、遠隔手続呼出し記述ファイルには、遠隔手続名と遠隔手続の引数の型や数、返り値の型が定義されている。たとえば図５において遠隔手続呼出し記述ファイルの例として、遠隔手続名がprintmessageで、引数はstringの型で１個を持ち、返り値はint型を持つことが示されている。本実施例では、予めクライアントプロセス実行部２０３で実行されるクライアントプロセスには第１のソースコードデバッグ部２０６が結合されているものとする。しかしながら、クライアントプロセスが起動された後で、クライアントプロセスのプロセス番号を知り第１のソースコードデバッグ部２０６を接続してもよい。第１のソースコードデバッグ部２０６は通信部２３における第１のＴＣＰソケット２３１による通信チャネルを介して中央デバッグ部２２のデバッガインタフェース２２２と結合している。この第１のＴＣＰソケット２３１は、具体的には、第１の衛星デバッグ部２０側に接続されたＴＣＰソケットと、中央デバッグ部２２側に接続されたＴＣＰソケットと、それらのＴＣＰソケット間を接続する通信線とからなっている。通信部２３の他のソケットやアクセス部についても同様に通信線とその両端部のインタフェース部からなっている。

【０００８】デバッグをする者は中央デバッグ部２２のディスプレイ端末装置２２３からコマンドを発行し、そのコマンドはデバッガインタフェース２２２を通じて第１のソースコードデバッグ部２０６に送られる。第１のソースコードデバッグ部２０６は、クライアントソースファイル記憶部２０１のクライアントソースファイルを参照して、ソースコードのデバッグを行う。第１のソースコードデバッグ部２０６に与えられたコマンドの実行結果は、第１のＴＣＰソケット２３１による通信チャネルを介してデバッガインタフェース２２２に送られディスプレイ端末装置２２３に表示される。

【０００９】そこで例えばブレークポイントで停止しているクライアントプロセスに対し、ソースコードについて１ステップずつの実行を指示するステップ実行コマンドを発行すると、第１のソースコードデバッグ部２０６は、クライアントプロセス実行部２０３でのステップ実行をさせ、クライアントプロセスが再度停止すると、ソースコード中での現在の停止地点を第１のブレークポイント検出部２２４に出力する。

【００１０】第１のブレークポイント検出部２２４は、第１のリモートファイルアクセス部２３２を介して、クライアントプロセスのソースコード中での現在の停止地点からクライアントソースファイル記憶部２０１のクライアントソースファイルを参照し、現在の停止地点が手続呼出しであるかどうかを調べ、手続呼出しでない場合は、デバッガインタフェース２２２には何も出力しない。また、現在の停止地点が手続呼出しであった場合は、遠隔手続呼出し記述ファイル記憶部２２６のファイルを参照し、手続呼出しの手続名と一致する遠隔手続名が存在すれば、現在の停止地点はクライアントプロセス中で遠隔手続呼出しを行う行であるとみなし、デバッガインタフェース２２２に遠隔手続名と遠隔手続に用いられる通信チャネルのチャネル番号と遠隔手続が起動されることを通知する。通信チャネルは、本実施例ではインタフェースとして第３のＴＣＰソケット２３３が用いられ、例えば通信チャネルのチャネル番号としてはＲＦＣ1060（1990年3月、J.Reynolds J.Pstel ISI）に規定されているインターネットプロトコルのポート番号が挙げられる。図３のクライアントのソースコードにおいて、クライアントプロセスが行７で「result=printmessage(&message,cl);」の実行をする前に停止しているとすれば、第１のブレークポイント検出部２２４は、図５に例示する遠隔手続呼出し記述ファイル記憶部２２６のファイルにおける行ｃの「int PRINTMESSAGE(string)= 1」と比較し、手続printmessageが遠隔手続PRINTMESSAGEに相当することを検出し、デバッガインタフェース２２２に遠隔手続printmessageが起動されることを通知する。デバッガインタフェース２２２はその第１のブレークポイント検出部２２４からの遠隔手続起動通知を記録する。

【００１１】次に、デバッグをする者が再度ステップ実行を指示すると、デバッガインタフェース２２２は、第

１のソースコードデバッグ部２０６に対して現在停止している「行８」の実行が終了したあと次に実行される行として「行９」に第１のブレークポイントを設定するよう通知する。また、デバッガインタフェース２２２は、記録された前記遠隔手続の起動通知の有無を調べ、通知の記録があれば、遠隔手続を含むプロセスを有する計算機名を調べる。この計算機名は、例えば、通知された遠隔手続名の呼出しに与えられる引数の中に前記遠隔手続を含むプロセスを有する計算機名が含まれているような遠隔手続の方式が用いられている場合は、第１のソースコードデバッグ部２０６にステップ実行を指示する代りに、手続呼出しの引数の詳細な内容を表示するコマンドを与え、その表示結果を調べることにより知ることができる。あるいは、遠隔手続の起動において、特定の計算機名を指定せずに遠隔手続投入先計算機決定部２２７によってその遠隔手続を含むプロセスを有する複数の計算機の中からひとつの計算機を決定するようにしてもよい。この場合は、デバッガインタフェース２２２はその遠隔手続投入先計算機決定部２２７からの出力により、遠隔手続を含むプロセスを有する計算機名を知ることができる。

【００１２】デバッガインタフェース２２２は前記遠隔手続を含むプロセスを有する計算機上に第２のソースコードデバッグ部２１７を起動し、第２のＴＣＰソケット２３４による通信チャネルを介し、その起動した第２のソースコードデバッグ部２１７と接続する。次に、デバッガインタフェース２２２はプロセステーブルアクセス部２３６を用い、通知されている遠隔手続に用いられる通信チャネルのチャネル番号から、前記遠隔手続を有し、クライアントプロセスからの接続を待っているプロセスのプロセス番号を特定し、第２のソースコードデバッグ部２１７に接続すべきサーバプロセスのプロセス番号として通知し、サーバプロセス実行部２１３と第２のソースコードデバッグ部２１７を接続する。デバッガインタフェース２２２は記録された前記遠隔手続の名前を第２のソースコードデバッグ部２１７に通知して、第２のブレークポイントを図４に例示するソースコードの「行ハ」に設定し、第３のブレークポイントを「行ホ」に設定する。

【００１３】次に、第２のソースコードデバッグ部２１７に対して継続実行を指示するコンティニュー命令を通知することで、サーバプロセス実行部２１３はクライアントプロセス実行部２０３からの接続を待ち、第２のソースコードデバッグ部２１７は第２のブレークポイントに到達するまで継続実行する。その後、デバッガインタフェース２２２は、第１のソースコードデバッグ部２０６に対して継続実行を指示するコンティニュー命令を通知する。クライアントプロセスは遠隔手続の実行が終了し、第１のブレークポイントに到達するまで実行を続ける。その後、デバッガインタフェース２２２はディスプレイ端末装置２２３からの入力が主に第２のソースコードデバッグ部２１７に送られ、第２のソースコードデバッグ部２１７から出力された結果はディスプレイ端末装置２２３に出力されるように切り替られる。

【００１４】次に、遠隔手続がクライアントプロセス実行部２０３で実行中のプロセスから発行されると、サーバプロセス実行部２１３のサーバプロセスは第２のブレークポイントで停止する。このように、第１のソースコードデバッグ部２０６と第２のソースコードデバッグ部２１７を制御することで、デバッグをする者には、遠隔手続呼出しが発行されたときの通信手順や引数の通信部分は見えなくて、あたかもクライアントプロセスから単なるシングルプロセス上の遠隔手続ではない手続呼出しが実行されたかのように見える。デバッグする者は特にサーバプロセスのデバッグをしていることを意識せずに遠隔手続をデバッグできる。

【００１５】遠隔手続のデバッグが終了し、例えばデバッグをする者がコンティニュー実行を指示すれば、デバッガインタフェース２２２はコンティニュー実行が指示されたことを記録し、サーバプロセス実行部２１３は「行ホ」において第３のブレークポイントで停止し、デバッガインタフェース２２２は第２のソースコードデバッグ部２１７から第２のＴＣＰソケット２３４を通じてそれを知った後、第２のソースコードデバッグ部２１７に第２のブレークポイントと第３のブレークポイントを取り除くよう指示する。第２のブレークポイントと第３のブレークポイントが取り除かれたら、デバッガインタフェース２２２は、ディスプレイ端末装置２２３からのコマンドが第１のソースコードデバッグ部２０６に送られるように切り替え、第１のソースコードデバッグ部２０６の出力がディスプレイ端末装置２２３に表示されるように切り替える。

【００１６】その後、デバッガインタフェース２２２は第２のソースコードデバッグ部２１７にサーバプロセス実行部２１３のサーバプロセスのデバッグを終了するように指示する。その結果、サーバプロセス実行部２１３は第２のソースコードデバッグ部２１７との接続が切断され、継続実行される。サーバプロセス実行部２１３からの遠隔手続の返り値がクライアントプロセス実行部２０３に戻され、第１のブレークポイントでクライアントプロセスが一旦停止するが、ソースコードデバッグ部２０６は、デバッグをする者がサーバプロセスの遠隔手続においてコンティニュ実行を指示したことが記録されているため、第１のブレークポイントを取り除きコンティニュ実行を第１のソースコードデバッグ部２０６に指示する。このように第１のソースコードデバッグ部２０６と第２のソースコードデバッグ部２１７を制御することで、デバッグをする者には、遠隔手続呼出しが終了したときの通信手順や返り値の通信部分は見えなくて、あたかもシングルプロセス上の、遠隔手続ではない手続から

返り値が戻されたかのように見える。従って、デバッグをする者は特にサーバプロセスのデバッグをしていることを意識せずに遠隔手続をデバッグすることができる。なお、以上に説明した実施例では、１つのクライアントプロセスと１つのサーバプロセス間の遠隔手続呼出しを例にとり本発明のデバッグ方式を説明したが、一般的には、一つのクライアントプロセスと並列に接続された複数のサーバプロセスからなる分散型の情報処理システムにおける任意のクライアントプロセスとサーバプロセス間のデバッグに本発明のデバッグ方式を適用できることは言うまでもない。また、クライアントプロセスにサーバプロセスが接続され、さらにそのサーバプロセスがクライアントプロセスとして別のサーバプロセスに接続されているような関係を含むシステムのデバッグにも本発明のデバッグ方式は同様に適用できる。

【００１７】

【発明の効果】以上述べたように、本発明によれば、サーバプロセスとクライアントプロセスの衛星デバッグ手段を適切に制御することにより、デバッグをする者には、遠隔手続呼出しが発行されたときの通信手順や引数の通信部分、あるいは、遠隔手続呼出しが終了したときの通信手順や返り値の通信部分は見えなくて、あたかもクライアントプロセスから単なるシングルプロセス上の、遠隔手続ではない手続呼出しが実行され、返り値が返されたかのように見える。従って、デバッグをする者は、従来の単一プロセスに対するデバッグ手法を用いることができ、プロセスの並行・並列動作を意識することなく簡単に遠隔手続呼出しを行う並行・並列分散プログラムのデバッグをすることができる。

【図面の簡単な説明】

【図１】本発明の構成を示すブロック図、

【図２】本発明の実施例を示すブロック図、

【図３】遠隔手続呼出しを行うクライアントプロセスのソースコードの例を示す図

【図４】サーバプロセスののソースコードの例を示す図

【図５】遠隔手続呼出し記述ファイルの例を示す図。

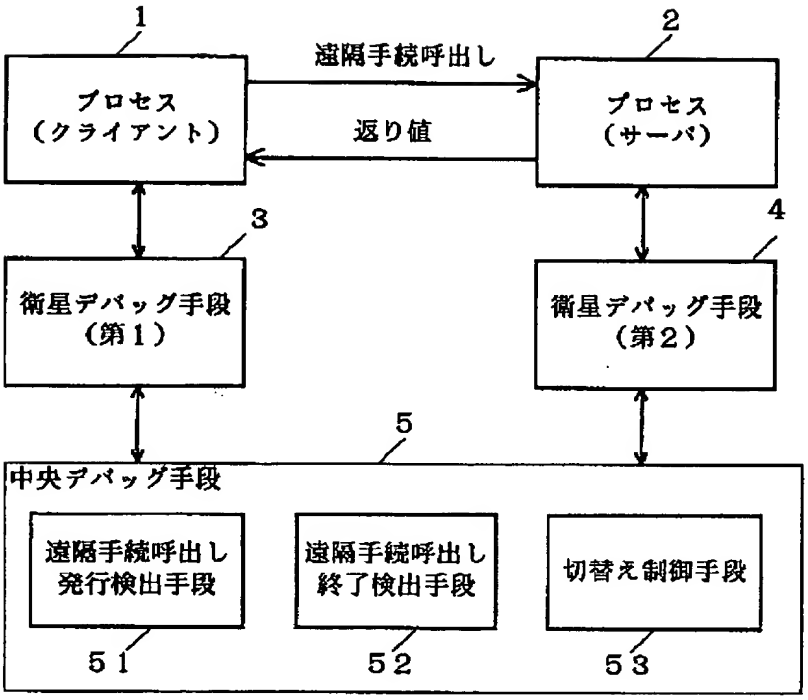
【符号の説明】

１，２…プロセス、３，４…衛星デバッグ手段、５…中央デバッグ手段、５１…遠隔手続呼出し発行検出手段、５２…遠隔手続呼出し終了検出手段、５３…切替え制御手段、２０…第１の衛星デバッグ部、２０１…クライアントソースファイル記憶部、２０２…クライアントプロセス実行形式ファイル記憶部、２０３…クライアントプロセス実行部、２０４…データ記憶部、２０６…第１のソースコードデバッグ部、２１…第２の衛星デバッグ部、２１１…サーバソースファイル記憶部、２１２…サーバプロセス実行形式ファイル記憶部、２１３…サーバプロセス実行部、２１４…データ記憶部、２１７…第２のソースコードデバッグ部、２１８…プロセステーブル２１８、２２…中央デバッグ部、２２２…デバッガインタフェース、２２３…ディスプレイ端末装置、２２４…第１のブレークポイント検出部、２２５…第２のブレークポイント検出部、２２６…遠隔手続呼出し記述ファイル記憶部、２２７…遠隔手続投入先計算機決定部２２７、２３…通信部、２３１…第１のＴＣＰソケット、２３２…第１のリモートファイルアクセス部、２３３…第３のＴＣＰソケット、２３４…第２のＴＣＰソケット、２３５…第２のリモートファイルアクセス部、２３６…プロセステーブルアクセス部２３６。

【図３】

```
1 main(server,message){
2
3   cl=clnt_create(server,MESSAGEPROG,MESSAGEVERS);
4   if(cl==NULL){
5   exit(1);
6   }
7   result=printmessage(&message,cl);
8 }
```

【図１】



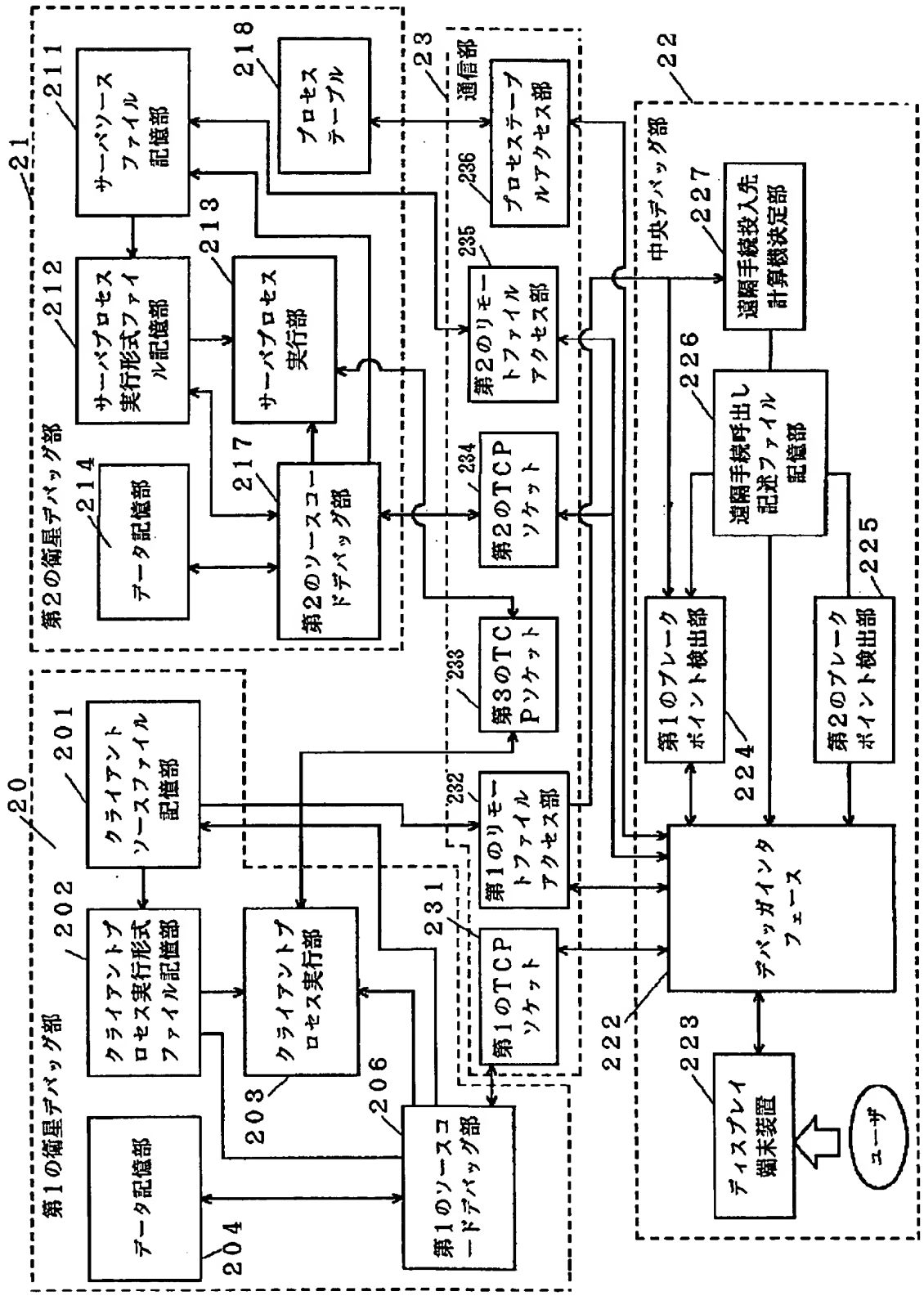
【図４】

```
1 printmessage(message){
2
3   printf("%s\n",*message);
4   result=1;
5   return(&result);
6 }
```

【図５】

```
a program MESSAGEPROG{
b   version MESSAGEVERS{
c     int PRINTMESSAGE(string)=1;
d   }=1;
e }=99;
```

【図２】


第１の衛星デバッグ部
20
クライアントソースファイル記憶部
201
クライアントプロセス実行形式ファイル記憶部
202
データ記憶部
204
クライアントプロセス実行部
203
第１のソースコードデバッグ部
206
第１のTCPソケット
231
第１のリモートファイルアクセス部
232
第３のTCPソケット
238
第２の衛星デバッグ部
21
サーバソースファイル記憶部
211
サーバプロセス実行形式ファイル記憶部
212
データ記憶部
214
サーバプロセス実行部
213
第２のソースコードデバッグ部
217
プロセステーブル
218
通信部
23
第２のTCPソケット
234
第２のリモートファイルアクセス部
235
プロセステーブルアクセス部
236
中央デバッグ部
22
第１のブレークポイント検出部
224
第２のブレークポイント検出部
225
遠隔手続呼出し記述ファイル記憶部
226
遠隔手続投入先計算機決定部
227
デバッガインタフェース
222
ディスプレイ端末装置
223
ユーザ